



# Methods

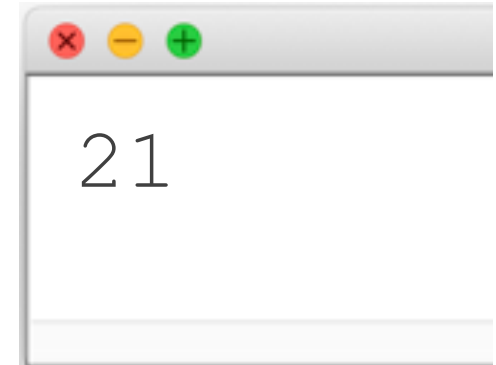
# Updating Variables

## Console Programs

```
int life = 42;  
life = 42 - life;  
life = 15;  
life = life / 2;  
println(life * 3);
```

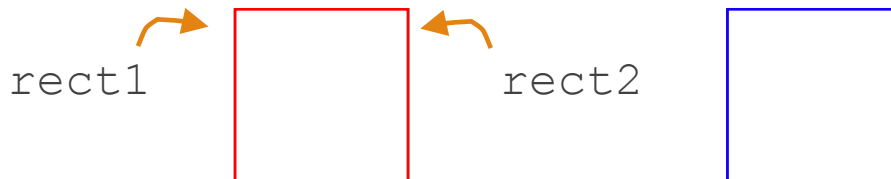
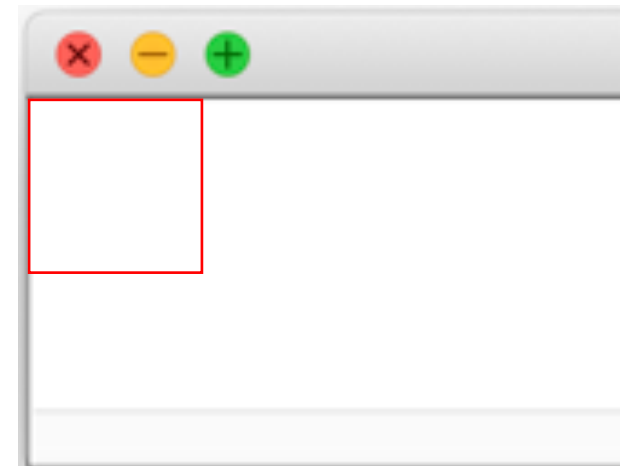
life

7



## Graphics Programs

```
GRect rectR = new GRect(100, 100);  
rectR.setColor(Color.RED);  
GRect rectB = new GRect(100, 100);  
rectB.setColor(Color.BLUE);  
rectB = rectR;  
add(rectB, 0, 0);
```



# So Many Boxes

```
int life = 42;
```

```
double d = 14.0 / 2.5;
```

```
String s = "This is a string";
```

```
GRect rect = new GRect(width, height);
```

```
GRect rect = new GRect(x, y);
```

We can create many types of variables in Java!!

# Animation loop

```
int count = 0;
GLabel countDisplay = new GLabel("" + count);
add(countDisplay, 1, 50);
while(true) {
    // updates text of label
    countDisplay.setLabel("" + count);
    count += 1;

    /* What happens when we insert
       * the code from cases 1, 2, and 3? */
}
```

```
(1) if (count > 10) {
    break;
}
pause(500);

(2) // nothing

(3) pause(500);
```



# Animation loop

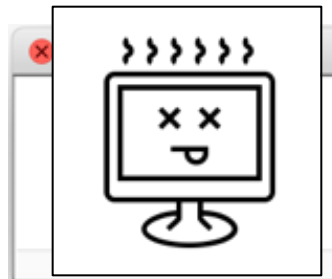
```
int count = 0;
GLabel countDisplay = new GLabel("" + count);
add(countDisplay, 1, 50);
while(true) {
    // updates text of label
    countDisplay.setLabel("" + count);
    count += 1;

    /* What happens when we insert
       * the code from cases 1, 2, and 3? */
}
```

(1) if (count > 10) {  
    break;  
}  
    pause(500);

(2) // nothing

(3) pause(500);



# Animation loop

```
int count = 0;
GLabel countDisplay = new GLabel("" + count);
add(countDisplay, 1, 50);
while(true) {
    // updates text of label
    countDisplay.setLabel("" + count);
    count += 1;

    /* What happens when we insert
       * the code from cases 1, 2, and 3? */
}
```

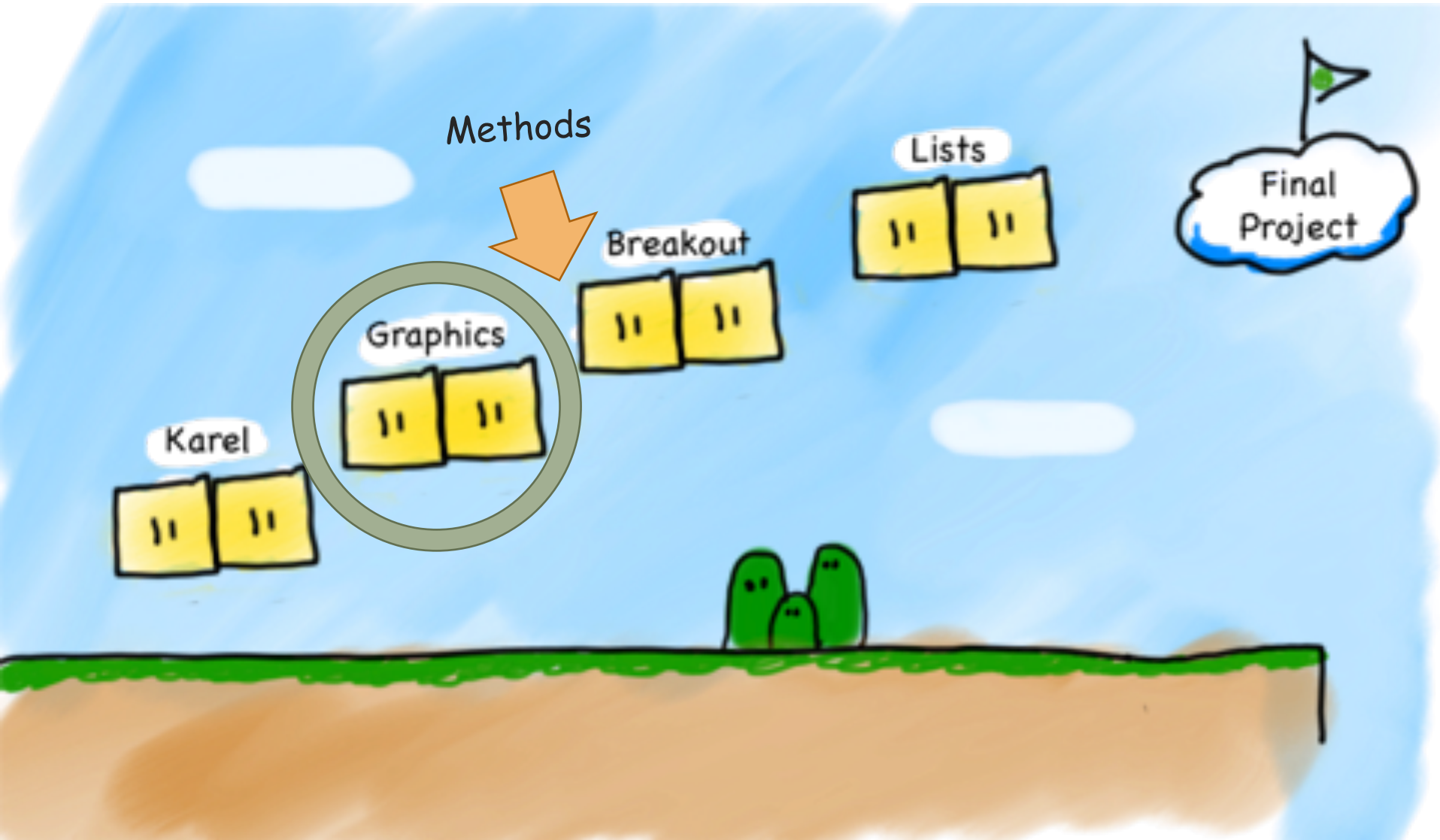
(1) if (count > 10) {  
    break;  
}  
    pause(500);

(2) // nothing

(3) pause(500);



# Our Second Step



# Today's Goals

1. What is a method and how do we talk about it?
2. How do we define our own methods?
3. What is happening when we call a method?





# Methods

```
turnRight();
```

```
move();
```

```
readInt("Int please! ");
```

```
println("hello world");
```

```
rect.getX();
```

```
drawRobotFace();
```

```
rect.setLocation(10, 20);
```

Today, we will learn exactly what these methods are doing!

# Defining a Method

```
private void turnRight() {  
    turnLeft();  
    turnLeft();  
    turnLeft();  
}
```



# Defining a Method

```
public void run() {  
    printAverage1();  
    printAverage2();  
}
```



```
private void printAverage1() {  
    double a = 5.0;  
    double b = 10.2;  
    double sum = a + b;  
    double mid = sum / 2;  
    println(mid);  
}
```

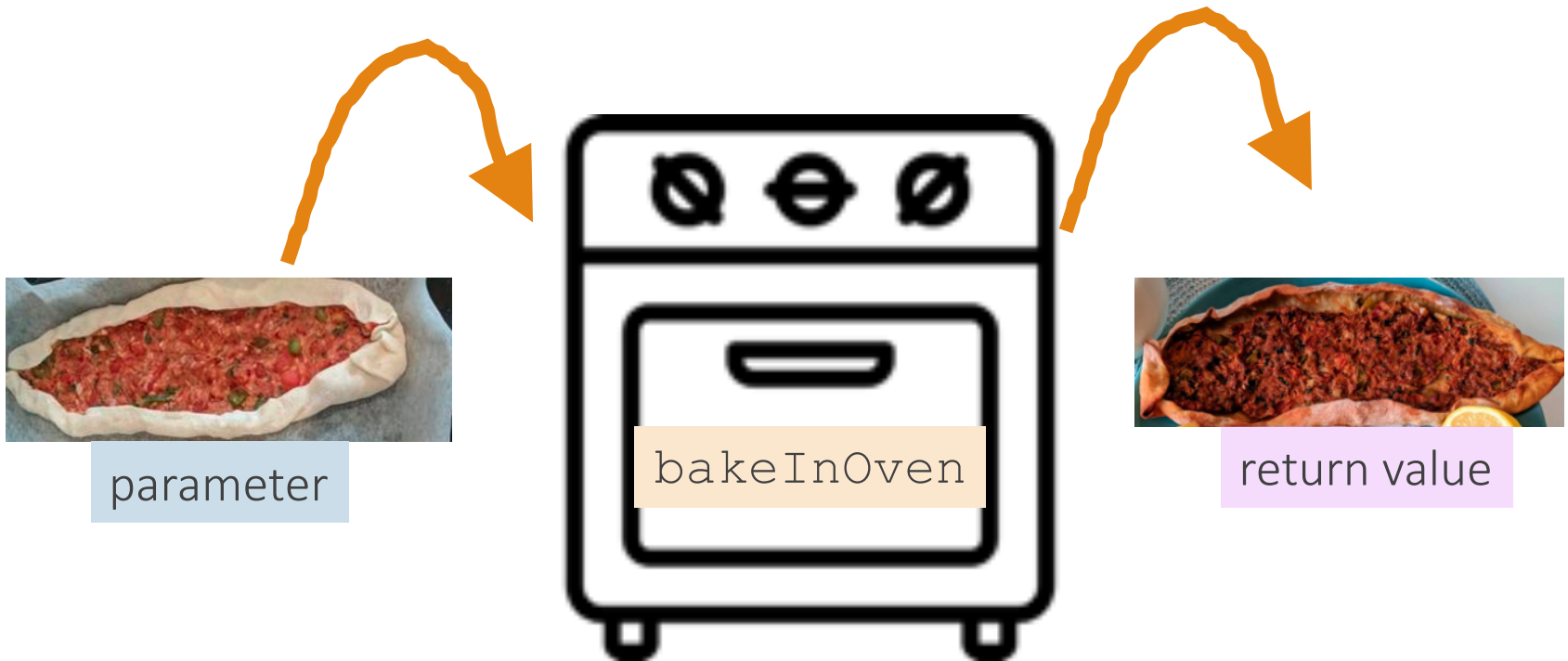
But wait...I thought  
methods help reuse code!

```
private void printAverage2() {  
    double a = 6; // int 6 → double 12.0  
    double b = 18.0;  
    double sum = a + b;  
    double mid = sum / 2;  
    println(mid);  
}
```



# Methods are Ovens

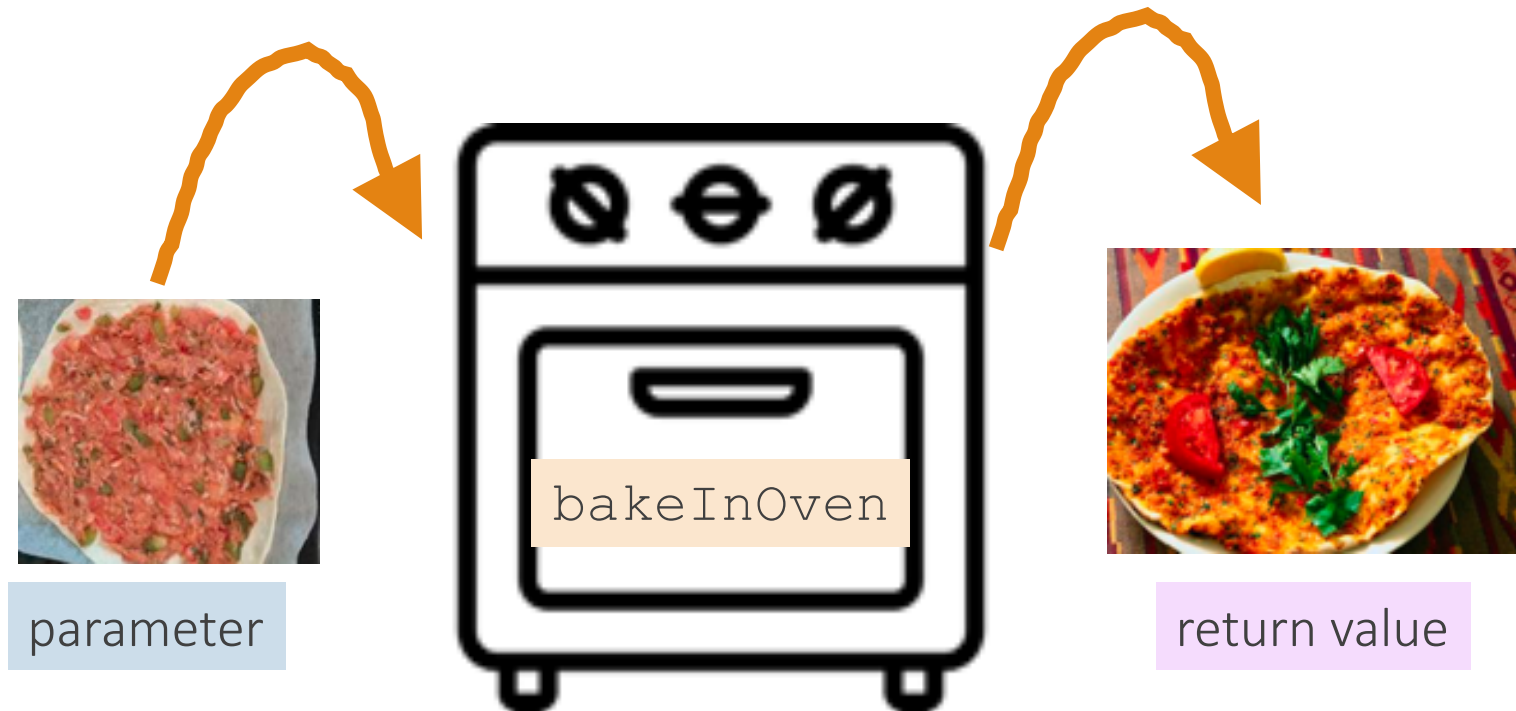
Java methods can take in data and return other data!!



# Ovens are Methods

Java methods can take in data and return other data!!

You don't need a different oven for lahmacun. Use the same one.



# Ovens are Methods

Java methods can take in data and return other data!!



Not all inputs work.

# The Java method

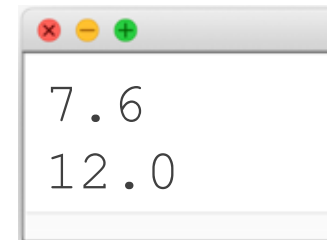
```
public void run() {  
    double mid1 = average(5.0, 10.2);  
    println(mid1);  
    double mid2 = average(6, 18);  
    println(mid2);  
}
```

method "call"

method name

```
private double average(double a, double b) {  
    double sum = a + b;  
    return sum / 2;  
}
```

method definition

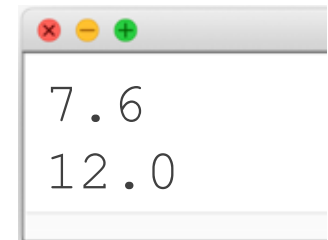


`average(double a, double b)` is a method that:

- Takes as input two **doubles** (a and b).
- Outputs a **double**
- Averages the two inputs.

# The Algebra Version

```
public void run() {  
    double mid1 = average(5.0, 10.2);  
    println(mid1);  
    double mid2 = average(6, 18);  
    println(mid2);  
}  
  
private double average(double a, double b) {  
    double sum = a + b;  
    return sum / 2;  
}
```



7.6  
12.0

Method definition:

$$\{ f(a, b) = (a + b) / 2$$

Method calls:

$$\begin{aligned} &\Rightarrow f(5.0, 10.2) = 7.6 \\ &\Rightarrow f(6, 18) = 12.0 \end{aligned}$$



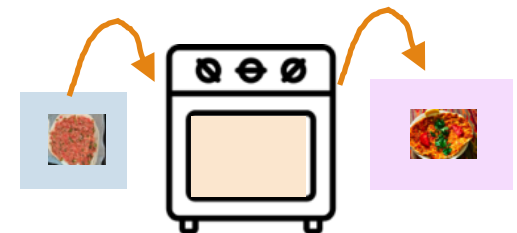
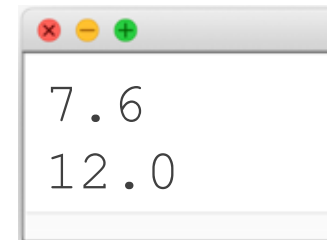
# The Java method

```
public void run() {  
    double mid1 = average(5.0, 10.2);  
    println(mid1);  
    double mid2 = average(6, 18);  
    println(mid2);  
}
```

Return type

Parameters

```
private double average(double a, double b) {  
    double sum = a + b;  
    return sum / 2;  
}
```

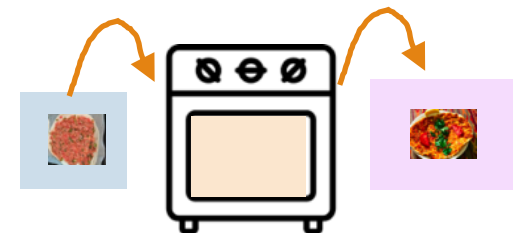
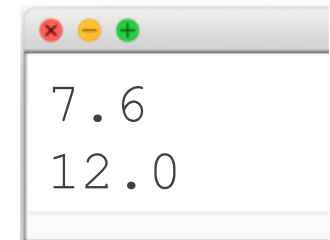


# Anatomy of a method

```
public void run() {  
    double mid1 = average(5.0, 10.2);  
    println(mid1);  
    double mid2 = average(6, 18);  
    println(mid2);  
}  
  
private double average(double a, double b) {  
    double sum = a + b;  
    return sum / 2;  
}
```

method body

return value

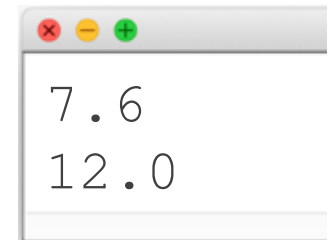


# Calling and Defining Methods

```
public void run() {  
    double mid1 = average(5.0, 10.2);  
    println(mid1);  
    double mid2 = average(6, 18);  
    println(mid2);  
}  
  
private double average(double a, double b) {  
    double sum = a + b;  
    return sum / 2;  
}
```

method definition

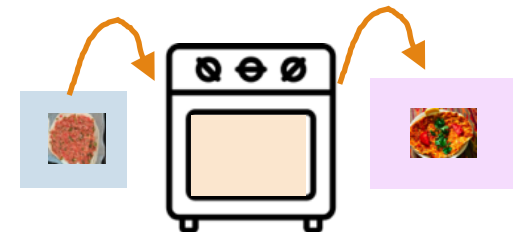
method "call"



arguments: calling (with actual **int** values)

vs

parameters: defining method input (any **int**)



# Explaining the `void` and the `()`

```
public void run() {  
    printIntro();  
}  
  
return type    name    parameters  
private void printIntro() {  
    println("Welcome to class");  
    println("It's the best part of my day.");  
    // nothing here  
}
```



**void** methods  
don't need a **return**.

`printIntro()` is a method that:

- Takes no parameters.
- Returns nothing.
- It just always prints: `Welcome to class`  
`It's the best part of my day.`

# Methods Dear to Our Heart

## Method name

## Parameter Types?

## Return Types?

```
average(5.0, 10.2);  
printIntro();
```

double, double  
(nothing)

double  
(nothing)

```
turnRight();
```

(nothing)

void

```
readInt("Enter age: ");  
println("You're cool!");
```

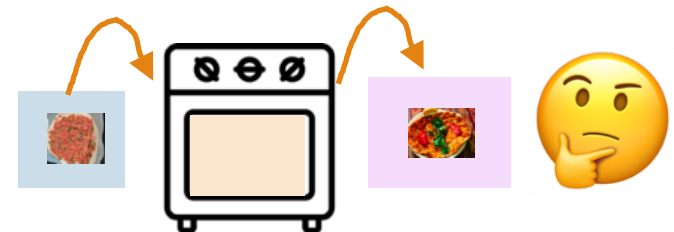
String  
String

int  
void

```
getWidth();  
rect.setLocation(10, 20);
```

(nothing)  
double, double

double  
void



# Questions?

TL;DR: (too long; don't read)  
(means the summary of what just happened)



parameter



return value

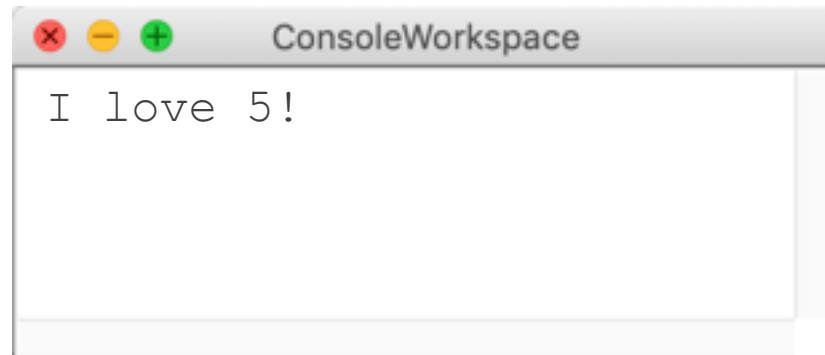
# Today's Goals

- ✓ 1. What is a method and how do we talk about it?
- 2. How do we define our own methods?
- 3. What is happening when we call a method?



# Parameter Example

```
public void run() {  
    printOpinion(5);  
}  
  
private void printOpinion(int num) {  
    if(num == 5) {  
        println("I love 5!");  
    } else {  
        println("Whatever");  
    }  
}
```





# Multiple Returns are OK

```
private String getMonthName(int index) {  
    if (index == 0) {  
        return "January";  
    }  
    if (index == 1) {  
        return "February";  
    }  
    ...  
    return "Unknown";  
}
```

getMonthName(0)?

returns

"January"

getMonthName(1)?

returns

"February"

getMonthName(200)?

returns

"Unknown"

# Multiple Returns are OK, but...



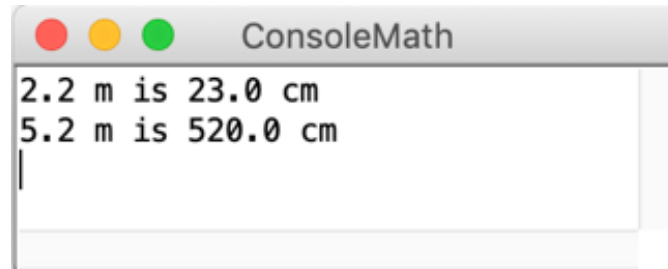
```
private String getMonthName(int index) {  
    if (index == 0) {  
        return "January";  
    }  
    if (index == 1) {  
        return "February";  
    }  
    ...  
    // return "Unknown";  
}
```

For all possible  
arguments of this type,  
*something* must be returned!

This method  
must return a  
result of  
type String



# Parameter + Returns



```
public void run() {
    double conversion = metersToCm(2.2);
    println("2.2 m is " + conversion + " cm");
    println("5.2 m is " + metersToCm(5.2) + " cm");
}

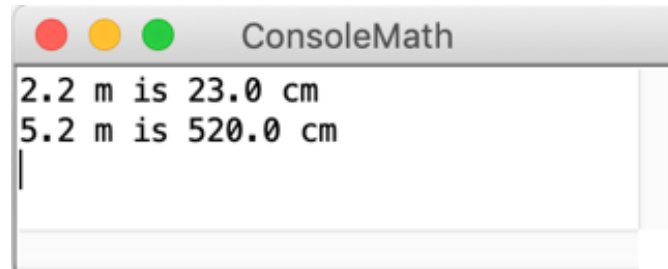
private ?????? metersToCm(???????) {
    /* Fill this in too */
}
```

} Step (1)

} Step (2)



# Parameter + Returns



```
public void run() {
    double conversion = metersToCm(2.2);
    println("2.2 m is " + conversion + " cm");
    println("5.2 m is " + metersToCm(5.2) + " cm");
}

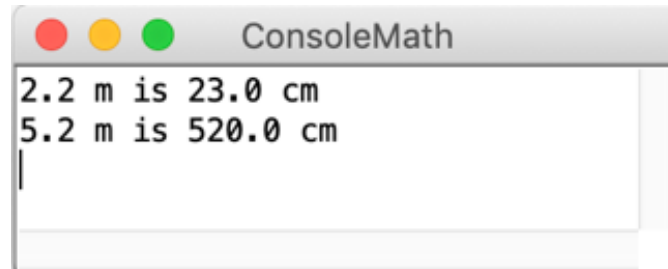
private double metersToCm(double meters) {

    return meters * 100;

}
```

! You must  
name your input  
variables!

# Parameter + Returns



```
public void run() {  
    double conversion = metersToCm(2.2);  
    println("2.2 m is " + conversion + " cm");  
    println("5.2 m is " + metersToCm(5.2) + " cm");  
}
```

```
private double metersToCm(double meters) {  
  
    return meters*100;  
  
}
```

⚠ Any non-**void** method  
must **return** something!

# Summary: Defining a Method

```
visibility type nameOfMethod(parameter types and names) {  
    statements  
}
```

- **visibility**: usually **private** or **public**
- **type**: type returned by method
  - **int**, **double**, etc. must include a **double** value!
  - Can be **void** to indicate that nothing is returned
- Input **parameters**: information passed into method
  - Must declare variable type AND variable name! (like **double** meter)
  - Can be empty ()

# Today's Goals

- ✓ 1. What is a method and how do we talk about it?
- ✓ 2. How do we define our own methods?
- ③ 3. What is happening when we call a method?



# Java Execution of Methods

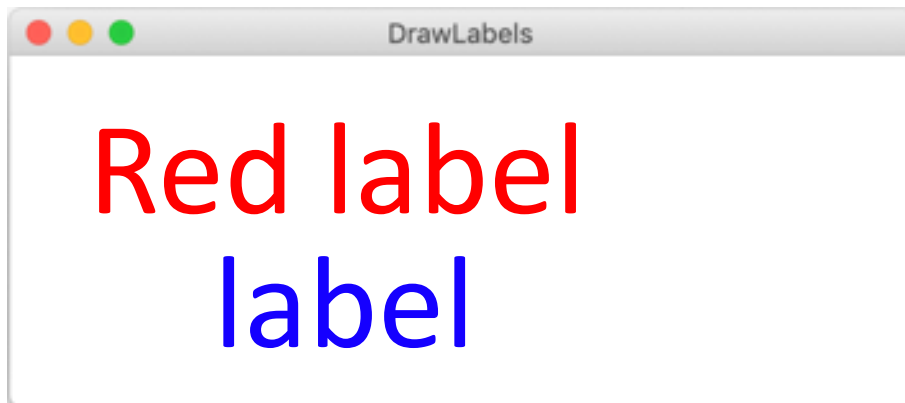


“equals”    (1) Evaluate right hand side  
=            (2) Store result in variable on left hand side



```
public void run() {  
    GLabel redLabel = coloredLabel("Red label", Color.RED);  
    add(redLabel, 50, 50);  
    GLabel label = coloredLabel("label", Color.BLUE);  
    add(label, 100, 100);  
}
```

```
private GLabel coloredLabel(String text, Color fill) {  
    GLabel label = new GLabel(text);  
    label.setFont("Calibri-50");  
    label.setColor(fill);  
    return label;  
}
```



What happens when  
we run this program?




```
public void run() {  
    GLabel redLabel = coloredLabel("Red label", Color.RED);  
    add(redLabel, 50, 50);  
    GLabel label = coloredLabel("label", Color.BLUE);  
    add(label, 100, 100);  
}
```

```
private GLabel coloredLabel(String text, Color fill) {  
    GLabel label = new GLabel(text);  
    label.setFont("Calibri-50");  
    label.setColor(fill);  
    return label;  
}
```




What happens when we run this program?




```
public void run() {  
    GLabel redLabel = coloredLabel("Red label", Color.RED);  
    add(redLabel, 50, 50);  
    GLabel label = coloredLabel("label", Color.BLUE);  
    add(label, 100, 100);  
}
```



- (1) Evaluate right hand side
- (2) Store result in variable  
on left hand side



```
public void run() {  
    GLabel redLabel = coloredLabel("Red label", Color.RED);  
    add(redLabel, 50, 50);  
    GLabel label = coloredLabel("label", Color.BLUE);  
    add(label, 100, 100);  
}
```



```
private GLabel coloredLabel(String text, Color fill) {  
    GLabel label = new GLabel(text);  
    label.setFont("Calibri-50");  
    label.setColor(fill);  
    return label;  
}
```



text


"Red label"

fill


Color.RED



- (1) Evaluate right hand side
- (2) Store result in variable on left hand side



```
public void run() {  
    GLabel redLabel = coloredLabel("Red label", Color.RED);  
    add(redLabel, 50, 50);  
    GLabel label = coloredLabel("label", Color.BLUE);  
    add(label, 100, 100);  
}
```



```
private GLabel coloredLabel(String text, Color fill) {  
    GLabel label = new GLabel(text);  
    label.setFont("Calibri-50");  
    label.setColor(fill);  
    return label;  
}
```



text

"Red label"


fill

Color.RED


label



Red label



```
public void run() {  
    GLabel redLabel = coloredLabel("Red label", Color.RED);  
    add(redLabel, 50, 50);  
    GLabel label = coloredLabel("Blue label", Color.BLUE);  
    add(label, 100, 100);  
}
```



```
private GLabel coloredLabel(String text, Color fill) {  
    GLabel label = new GLabel(text);  
    label.setFont("Calibri-50");  
    label.setColor(fill);  
    return label;  
}
```



text

"Red label"


fill

Color.RED

label



Red label




```
public void run() {  
    GLabel redLabel = coloredLabel("Red label", Color.RED);  
    add(redLabel, 50, 50);  
    GLabel label = coloredLabel("label", Color.BLUE);  
    add(label, 100, 100);  
}
```



```
private GLabel coloredLabel(String text, Color fill) {  
    GLabel label = new GLabel(text);  
    label.setFont("Calibri-50");  
    label.setColor(fill);  
    return label;  
}
```



redLabel  
└─→ Red label



```
public void run() {  
    GLabel redLabel = coloredLabel("Red label", Color.RED);  
    add(redLabel, 50, 50);  
    GLabel label = coloredLabel("label", Color.BLUE);  
    add(label, 100, 100);  
}
```




```
private GLabel coloredLabel(String text, Color fill) {  
    GLabel label = new GLabel(text);  
    label.setFont("Calibri-50");  
    label.setColor(fill);  
    return label;  
}
```




redLabel  
└─→ Red label





```
public void run() {  
    GLabel redLabel = coloredLabel("Red label", Color.RED);  
    add(redLabel, 50, 50);  
    GLabel label = coloredLabel("label", Color.BLUE);  
    add(label, 100, 100);  
}
```



```
private GLabel coloredLabel(String text, Color fill) {  
    GLabel label = new GLabel(text);  
    label.setFont("Calibri-50");  
    label.setColor(fill);  
    return label;  
}
```



text

"label"

fill


Color.BLUE




redLabel



Red label



```
public void run() {  
    GLabel redLabel = coloredLabel("Red label", Color.RED);  
    add(redLabel, 50, 50);  
    GLabel label = coloredLabel("label", Color.BLUE);  
    add(label, 100, 100);  
}
```



```
private GLabel coloredLabel(String text, Color fill) {  
    GLabel label = new GLabel(text);  
    label.setFont("Calibri-50");  
    label.setColor(fill);  
    return label;  
}
```



text

"label"

fill

Color.BLUE

label




label


redLabel



Red label



```
public void run() {  
    GLabel redLabel = coloredLabel("Red label", Color.RED);  
    add(redLabel, 50, 50);  
    GLabel label = coloredLabel("label", Color.BLUE);  
    add(label, 100, 100);  
}
```



```
private GLabel coloredLabel(String text, Color fill) {  
    GLabel label = new GLabel(text);  
    label.setFont("Calibri-50");  
    label.setColor(fill);  
    return label;  
}
```



text

"label"

fill

Color.BLUE

label




label

redLabel



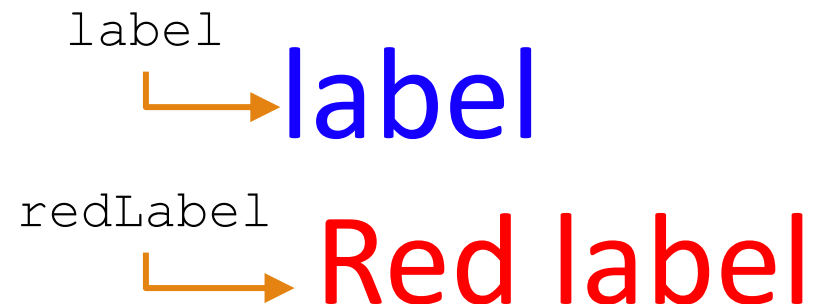
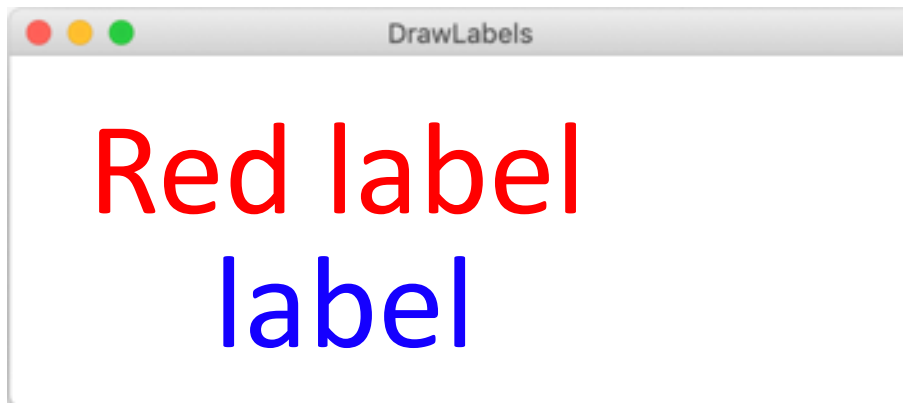
Red label



```
public void run() {  
    GLabel redLabel = coloredLabel("Red label", Color.RED);  
    add(redLabel, 50, 50);  
    GLabel label = coloredLabel("label", Color.BLUE);  
    add(label, 100, 100);  
}
```



```
private GLabel coloredLabel(String text, Color fill) {  
    GLabel label = new GLabel(text);  
    label.setFont("Calibri-50");  
    label.setColor(fill);  
    return label;  
}
```



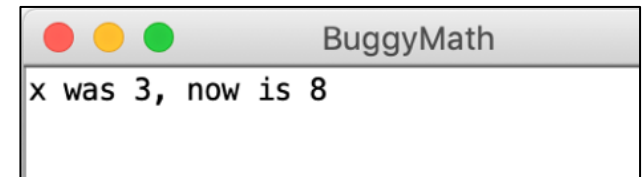
Questions?

More Examples

# Bad Times with Methods

!   
`public void run() {  
 int x = 3;  
 int prevX = x;  
 addFive(x);  
 println("x was " + prevX + ", now" + x);  
}`

!   
`private void addFive(int x) {  
 x += 5;  
 println(x); !  
}`



(intention)

There are three bugs in this program!



# Good Times with Methods

```
public void run() {  
    int x = 3;  
    int prevX = x;  
    x = addFive(x);  
    println("x was " + prevX + ", now" + x);  
}  
  
private int addFive(int x) {  
    x += 5;  
    return x;  
}
```

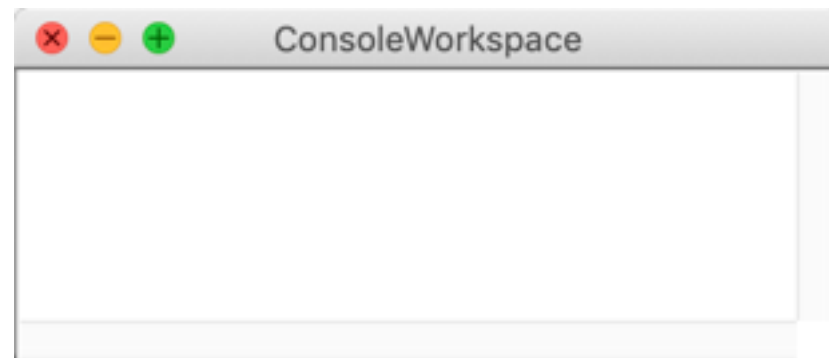
That's more like it!



# Changed Name

```
private void run() {  
    int num = 5;  
    cow(num);  
}
```

```
private void cow(int grass) {  
    println(grass);  
}
```



# Changed Name

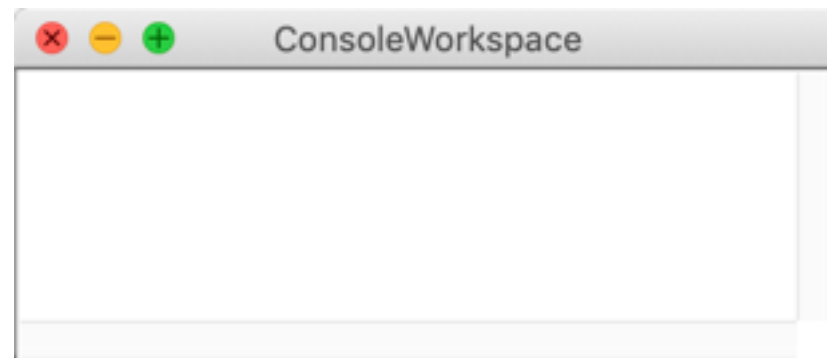


```
private void run() {  
    int num = 5;  
    cow(num);  
}
```

num

5

```
private void cow(int grass) {  
    println(grass);  
}
```



# Changed Name



```
private void run() {  
    int num = 5;  
    cow(num);  
}
```

num

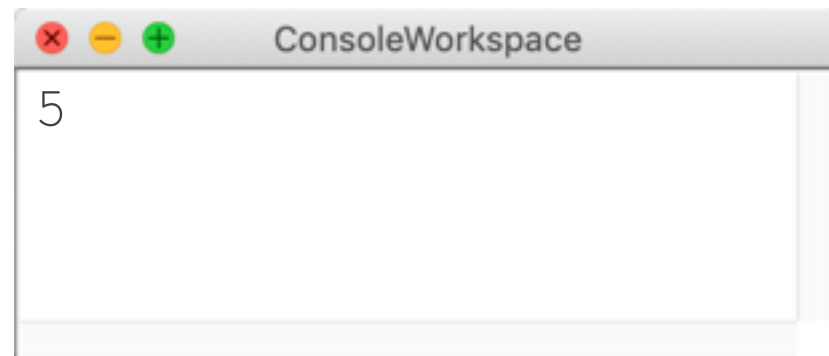
5



```
private void cow(int grass) {  
    println(grass);  
}
```

grass

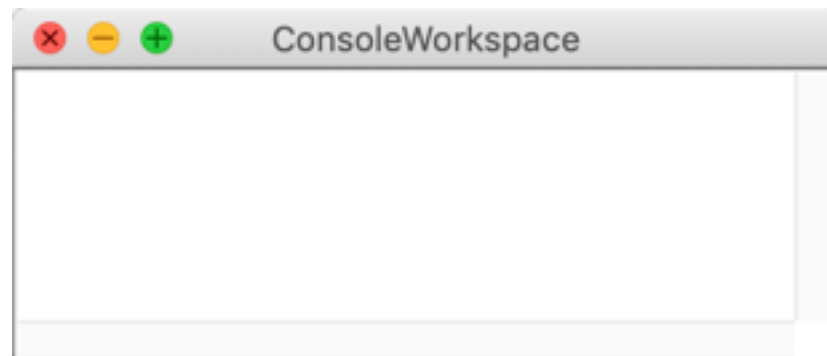
5



# Same Variable

```
private void run() {  
    int num = 5;  
    cat();  
}
```

```
private void cat() {  
    int num = 10;  
    println(num);  
}
```



# Same Variable



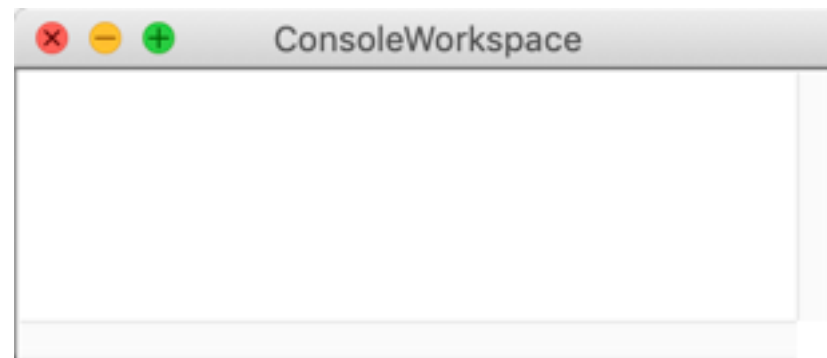
```
private void run() {  
    int num = 5;  
    cat();  
}
```

num

5



```
private void cat() {  
    int num = 10;  
    println(num);  
}
```



# Same Variable



```
private void run() {  
    int num = 5;  
    cat();  
}
```

num

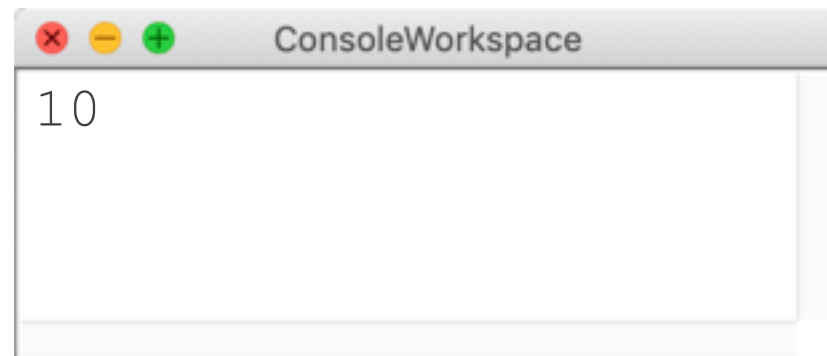
5



```
private void cat() {  
    int num = 10;  
    println(num);  
}
```

num

10



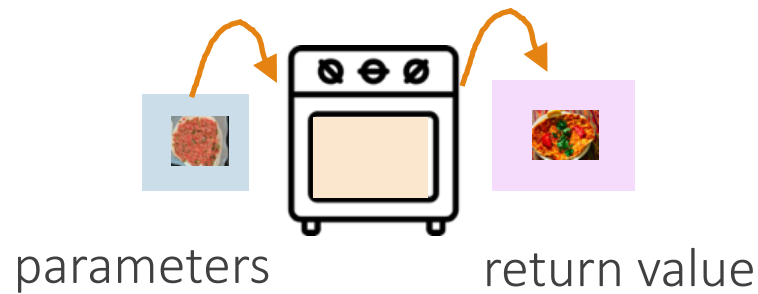
# Today's Goals

- ✓ 1. What is a method and how do we talk about it?
- ✓ 2. How do we define our own methods?
- ✓ 3. What is happening when we call a method?



# Review

A method:



```
private int addFive(int x) {  
    x += 5;  
    return x;  
}
```

If you declare a return type,  
you must return a value of that type.

```
private void cat() {  
    int num = 10;  
    println(num);  
}
```

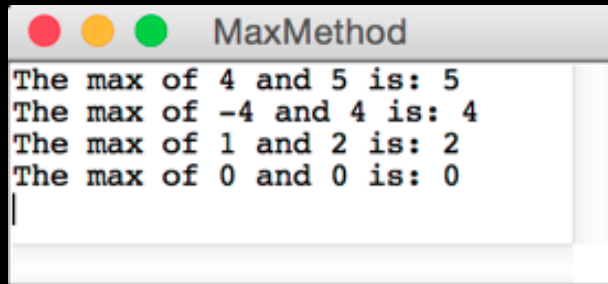
void: no return values  
(): no parameters.  
println() is NOT **return**!!!!



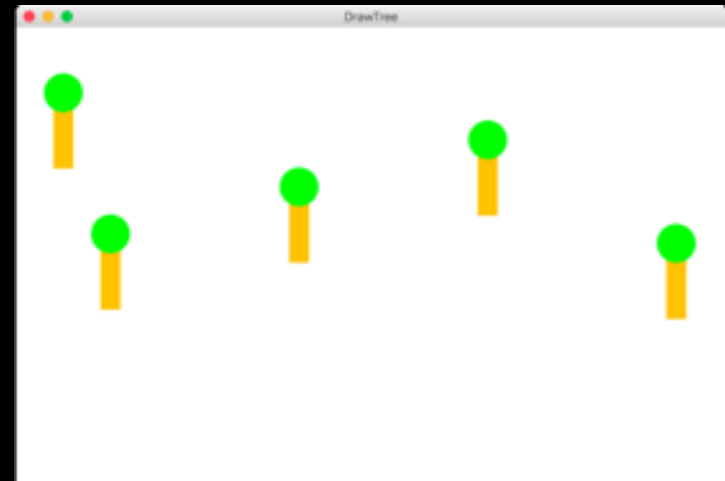
Today's material is *difficult*.

👍 Good job surviving 👍

👋 Bring your questions to section! 👋



```
MaxMethod
The max of 4 and 5 is: 5
The max of -4 and 4 is: 4
The max of 1 and 2 is: 2
The max of 0 and 0 is: 0
```



Mad Methods

# Boolean



# Boolean Variable

```
boolean karelIsAwesome = true;
```

```
boolean myBool = 1 < 2;
```

# Boolean Operations

```
boolean a = true;
```

```
boolean b = false;
```

```
//This is false
```

```
boolean a_and_b = a && b;
```

```
//This is true
```

```
boolean a_or_b = a || b;
```

```
// This is false
```

```
boolean not_a = !a;
```

**Now that's style!**

